Available online at www.sciencedirect.com
ScienceDirect
Nuclear Physics B 993 (2023) 116253
NUCLEAR PHYSICS B
www.elsevier.com/locate/nuclphysb

Balancing act: Multivariate rational reconstruction for IBP

A.V. Belitsky [a,*], A.V. Smirnov [b,c], R.V. Yakovlev [b,c]

[a] *Department of Physics, Arizona State University, Tempe, AZ 85287-1504, USA*
[b] *Research Computing Center, Moscow State University, 119992 Moscow, Russia*
[c] *Moscow Center for Fundamental and Applied Mathematics, 119992 Moscow, Russia*

Received 14 March 2023; received in revised form 24 May 2023; accepted 3 June 2023
Available online 16 June 2023
Editor: Stephan Stieberger

Abstract

We address the problem of unambiguous reconstruction of rational functions of many variables. This is particularly relevant for recovery of exact expansion coefficients in integration-by-parts identites (IBPs) based on modular arithmetic. These IBPs are indispensable in modern approaches to evaluation of multiloop Feynman integrals by means of differential equations. Modular arithmetic is far more superior to algebraic implementations when one deals with high-multiplicity situations involving a large number of Lorentz invariants. We introduce a new method based on balanced relations which allows one to achieve the goal of a robust functional restoration with minimal data input. The technique is implemented as a Mathematica package `Reconstruction.m` in the FIRE6 environment and thus successfully demonstrates a proof of concept.

1. Introduction

Integration-by-parts identities (IBPs) [1], see also Chapter 6 of the book [2], are an indispensable tool to reduce an arbitrarily large set of Feynman integrals to a finite set [3] of the so-called

* Corresponding author.
E-mail address: andrei.belitsky@asu.edu (A.V. Belitsky).

https://doi.org/10.1016/j.nuclphysb.2023.116253

Master Integrals (MIs). With development of powerful computers, a number of programs had become available over the past twenty years to handle algebraically a plethora of IBPs. The bulk of them [4–8] (for the exception of [9]) is based on the renowned Laporta algorithm [10] of Gauss elimination for a given choice of priority ordering.

IBPs form a system of linear algebraic equations with matrix coefficients whose elements are generically given by rational functions

$$F(\boldsymbol{x}) = \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})}\,. \tag{1}$$

Here P and Q are polynomials of (typically) different degrees in the number of space-time dimension d and $(L-1)$ Lorentz invariants. These are cumulatively denoted by the vector of variables $\boldsymbol{x} = (x_1, \dots, x_L)$.

The main problem with the Laporta reduction is the *swelling of intermediate expressions* when one performs the elimination in an algebraic manner and which, in turn, leads to severe computer performance issues, as these become incredibly time consuming to manipulate with and hard to store in a memory. The final form of the coefficients is, however, rather concise in length (several orders of magnitude less than their intermediate form).

A natural way out would be to perform all calculations numerically and then attempt their analytic reconstruction. The use of floating point arithmetic is a no-starter in this context however since one is aiming at an exact reconstruction of the expansion coefficients. The use of rational numbers instead is plagued by problems akin to ones emerging in analytic treatments since they require arbitrary precision arithmetic which is time consuming. So an idea was put forward in Ref. [11] to use numerical techniques over finite fields in computer algebra manipulations of IBPs. These are obviously advantageous compared to the ones we just alluded to above since, as the name suggests, there is only a finite number of elements involved (with well-defined inverses) and they can be represented by machine-size integers. Their disadvantage, however, is information loss along the way and thus the necessity to use several finite fields for back recovery of rational numbers. However, these efforts are far less demanding than the direct use of the latter. This rational reconstruction from its images in several fields has been known for quite a while and is implemented through the Extended Euclidean Algorithm [12,13] which relies on the Chinese Remainder Theorem [14].

Having addressed the proper numbers for numerical calculations, the main problem then consists in the actual reconstruction of *the black box* (1) from sample data (aka black box points) with high probability of success. While univariate methods date back to more than a century ago as celebrated Newton and Thiele interpolations [15] for polynomial and rational functions, respectively, multivariate techniques are relatively new. Interpolations for sparse[1] functions of many variables were addressed in Refs. [16–23]. Reconstruction methods for dense[2] multivariate functions are more rare. A generalization of the Thiele method was proposed and implemented with the release of the FIRE6 IBP reduction framework in Ref. [5]. However, it can hardly be used beyond two variables, since it faces severe computational challenges. The most prominent sparse reconstructions are based on the so-called homogeneous interpolation [21,22] and the Zippel algorithm [24]. The former was adopted and remastered in Refs. [25] and [26] for scattering amplitude problems through the `FiniteFlow` and `FireFly` packages, respectively. `Fire-`

[1] It refers to a given upper bound on the number of terms in P and Q of Eq. (1).

[2] I.e., unconstrained number of terms in P and Q.

`Fly` also employs the Zippel algorithm and was recently integrated with Kira2.0 IBP program in Ref. [8].

In circumstances when the time of numerical computations of a single sample point is comparable to the one of the total reconstruction, it is very important to have a method which requires minimal initial data set. In this regard, the homogeneous multivariate reconstruction is indeed a viable tool, however, it requires sufficiently high number of probes for successful sought-after reconstruction as it is very sensitive to the total power of polynomials building up the numerator/denominator in Eq. (1). This technique will be used by us as a benchmark for the method which we propose in this work.[3] We will demonstrate that it will allow us for a more economical computational efforts compared to the former since it requires less black box probes.

Our subsequent presentation is organized as follows. In the next section, we recall classical methods of univariate interpolations. Then in Sect. 3, we present our new framework based on the so-called *balanced* reconstruction. We address the issue of the most optimal ordering in Sect. 4 and then, in Sect. 5, compare our method with the homogenous one. In Sect. 6, we introduce a Mathematica code integrated with the FIRE6 program for the balanced reconstruction of IBPs and give a thorough example in Sect. 7. Finally, we conclude and discuss future directions. This manuscript is also accompanied by the Mathematica script code `Reconstruction.m` and the notebook `reconstruction.nb` detailing all recovery steps, as well as syntax and commands, for a typical process of IBP reduction.

2. Classical univariate reconstructions

To start with, let us recall two classical results used in polynomial and rational interpolation of functions of a single variable x, which we build upon in the following sections. These are known as Newton and Thiele methods, respectively.

Throughout this paper, we will be adhering to the following notations: N will be the number of samples for the Newton method, while T will be the number of data points for the Thiele method. Greek letters will denote integer labels of fixed numerical values of variables, e.g., x_α.

2.1. Newton method

The most basic method of polynomial interpolation of a function $f(x)$ is based on the so-called Newton interpolating polynomials $f_N(x)$ on N distinct sampling data points x_α with $\alpha = 1, \dots, N$,

$$\begin{aligned} f_N(x) &= \text{Newton}_x[f(x), N] \\ &\equiv a_1 + (x - x_1)\Big[a_2 + (x - x_2)\big[a_3 + (x - x_3)\,[a_4 + \dots]\big]\Big]. \end{aligned} \tag{2}$$

The accompanying coefficients are defined recursively through the divided differences

$$\begin{aligned} a_1 &= f(x_1)\,, \\ a_2 &= [f(x_1), f(x_2)] \equiv \frac{f(x_1) - f(x_2)}{x_1 - x_2}\,, \\ a_3 &= [f(x_1), f(x_2), f(x_2)] \equiv \frac{[f(x_1), f(x_2)] - [f(x_2), f(x_3)]}{x_1 - x_3}\,, \end{aligned}$$

[3] A preliminary version of the method, which we introduce and explore in the current paper, was discussed in Ref. [27].

$$a_4 = [f(x_1), f(x_2), f(x_3), f(x_4)] \equiv \frac{[f(x_1), f(x_2), f(x_3)] - [f(x_2), f(x_3), f(x_4)]}{x_1 - x_4},$$

$$\vdots$$

$$\begin{aligned} a_N &= [f(x_1), f(x_2), f(x_3), \dots, f(x_N)] \\ &\equiv \frac{[f(x_1), f(x_2), f(x_3), \dots, f(x_{N-1})] - [f(x_2), f(x_3), f(x_4), \dots, f(x_N)]}{x_1 - x_N}. \end{aligned} \tag{3}$$

Obviously, if the function of interest $f(x)$ is known to be a polynomial of a predetermined degree deg$[f(x)]$ to start with, one can unambiguously reconstruct it by sampling in

$$N = \deg_x[f(x)] + 2 \tag{4}$$

points. The last one being the control probe such that the function does not change by adding more data points

$$f_{N+1}(x) = f_N(x) = f(x). \tag{5}$$

The advantage of the Newton method compared to the naive power expansion with unknown coefficients is the fact that an addition on a new data point does not necessitate reevaluation of all of the coefficients from scratch.

2.2. *Thiele method*

Rational interpolation of a function $f(x)$, which typically yields a better approximation than the above polynomial interpolation, is achieved with the help of the Thiele continued fraction on T black box probes x_α with $\alpha = 1, \dots, T$

$$\begin{aligned} f_T(x) &= \text{Thiele}_x[f(x), T] \\ &\equiv b_0 + (x - x_1)\left[b_1 + (x - x_2)\left[b_2 + (x - x_3)[b_4 + \dots]^{-1}\right]^{-1}\right]^{-1}, \end{aligned} \tag{6}$$

and the coefficients being determined by the following relations

$$b_1 = f(x_1),$$

$$b_2 = [f(x_1), f(x_2)]_r \equiv \frac{x_1 - x_2}{f(x_1) - f(x_2)},$$

$$b_3 = [f(x_1), f(x_2), f(x_2)]_r \equiv \frac{x_1 - x_3}{[f(x_1), f(x_2)]_r - [f(x_2), f(x_3)]_r},$$

$$b_4 = [f(x_1), f(x_2), f(x_3), f(x_4)]_r \equiv \frac{x_1 - x_4}{[f(x_1), f(x_2), f(x_3)]_r - [f(x_2), f(x_3), f(x_4)]_r},$$

$$\vdots$$

$$\begin{aligned} b_N &= [f(x_1), f(x_2), f(x_3), \dots, f(x_N)]_r \\ &\equiv \frac{x_1 - x_N}{[f(x_1), f(x_2), f(x_3), \dots, f(x_{N-1})]_r - [f(x_2), f(x_3), f(x_4), \dots, f(x_N)]_r}, \end{aligned} \tag{7}$$

closely related to the reciprocal differences.

Again, if a function $f(x)$ is known to be rational to begin with, the method allows to exactly reconstruct it by sampling in T points, with the latter determined by the following estimate (including the control probe)

$$T \simeq 2 \times \max \left\{ \deg_x[\text{Numerator}[f(x)]], \deg_x[\text{Denominator}[f(x)]] \right\} + 1 \,. \tag{8}$$

Akin to the Newton method, Thiele reconstruction does not require recalculation of all the b_j's with every new sample added, contrary to other methods, for instance, the so-called barycentric interpolation [28]. Sometimes, the above algorithm may yield a vanishing denominator, for instance, when two successive points possess the same dependent value or when one samples three collinear successive data points. In these circumstances, all one has to do is to perturb data points ever so slightly to get rid of the problem.

3. Balanced reconstruction

With the above lightning overview of univariate interpolations behind us, let us introduce a new approach to multivariate rational reconstruction, which we dub *the balanced reconstruction*.

Consider a rational multivariate function of L variables

$$F = F(\boldsymbol{x}) \,, \quad \text{with} \quad \boldsymbol{x} = (x_1, \dots, x_L) \,. \tag{9}$$

Let us spit the total vector of variables $\boldsymbol{x}$ into three orthogonal vector subspaces

$$\boldsymbol{x} = (\boldsymbol{d}, x_j, \boldsymbol{r}) \,, \quad \text{with} \quad \boldsymbol{d} = (x_1, \dots, x_{j-1}) \,, \quad \boldsymbol{r} = (x_{j+1}, \dots, x_L) \,, \tag{10}$$

with the $\boldsymbol{d}$-vector taking on the meaning of analytically reconstructed, or (d)one, variables, x_j being the variables under consideration and $\boldsymbol{r}$ being the (r)emainder. We designate the function $F(\boldsymbol{x})$ with $\boldsymbol{d}$ reconstructed variables as

$$F_{\boldsymbol{d}}(\boldsymbol{d}, x_j, \boldsymbol{r}) \,. \tag{11}$$

Before one starts the recovery algorithm, one has to get an estimate on the minimal number of sampling points needed for successful reconstruction. This is accomplished by performing the univariate Thiele restoration for each variables x_j from the vector $\boldsymbol{x}$ with all others $\boldsymbol{x}\backslash x_j$ kept fixed, yielding a value T_j for a stable reconstruction. These are then used to get the minimal number of sample data points needed

$$\{x_{1,\alpha}, \alpha = 1, \dots, T_1; \quad x_{2,\beta}, \beta = 1, \dots, N_2; \quad \dots \quad ; \quad x_{L,\gamma}, \gamma = 1, \dots, N_L\} \,, \tag{12}$$

with

$$N_j \simeq [T_j/2] \tag{13}$$

to be explained below (see Sect. 4).

The algorithm consists in the following steps.

1. Numerically compute values of the function F with fixed values of all variables in their respective ranges, determined from the preliminary estimates alluded to above,

$$x_{1,\alpha} \,, \quad \alpha = \{1, \dots, T_1\} \,, \qquad (x_2, \dots, x_L)_\beta \,, \quad \beta = \{1, \dots, N_{2,\dots,L}\} \,. \tag{14}$$

2. The first variable x_1 is reconstructed by means of the Thiele method,

$$F_{x_1}(x_1, \boldsymbol{r}_\beta) = \text{Thiele}_{x_1}[F(x_1, \boldsymbol{r}_\beta), T_1] \,. \tag{15}$$

3. Let $\boldsymbol{d}$ be the vector of already reconstructed variables [i.e., $\boldsymbol{d} = (x_1)$ after the first step]. Collect tables of the function

$$F_d(\boldsymbol{d}, x_{j,\alpha}, \boldsymbol{r}_\beta) \tag{16}$$

with $\alpha \in \{1, \dots, N_j\}$ and $\beta \in \{1, \dots, N_r\}$ from the above two steps. All other variables $\boldsymbol{x}\backslash x_1$ are handled by the *balanced* Newton method as follows.

4. Compute values of the function $F = F_d(\boldsymbol{d}_0, x_{j,\alpha}, \boldsymbol{r}_\beta)$ for a single fixed numerical value of the vector of already done variables $\boldsymbol{d} = \boldsymbol{d}_0$, T_j values of the variable x_j under reconstruction $x_{j,\alpha}$, $\alpha \in \{1, \dots, T_j\}$, and N_r values of the rest $\boldsymbol{r}_\beta$, $\beta \in \{1, \dots, N_r\}$ (same as in step 1). Thiele reconstructs x_j from the set $F_d(\boldsymbol{d}_0, x_{j,\alpha}, \boldsymbol{r}_\beta)$ obtaining

$$\textit{the balancing tables:} \qquad F_{d,x_j}(\boldsymbol{d}_0, x_j, \boldsymbol{r}_\beta) = F_d(\boldsymbol{d}_0, x_j, \boldsymbol{r}_\beta) \tag{17}$$

for the variable x_j. Notice that this reconstruction step is univariate in the variable x_j.

5. Balance the set of values $F_d(\boldsymbol{d}, x_{j,\alpha}, \boldsymbol{r}_\beta)$ computed in step 3 with the balancing tables from step 4 by evaluating

$$V(\boldsymbol{d}, x_j, \boldsymbol{r}_\beta) = \frac{F_d(\boldsymbol{d}, x_{j,\alpha}, \boldsymbol{r}_\beta) \times F_{d,x_j}(\boldsymbol{d}_0, x_j, \boldsymbol{r}_\beta)}{F_d(\boldsymbol{d}_0, x_{j,\alpha}, \boldsymbol{r}_\beta)} . \tag{18}$$

6. Factorize $V(\boldsymbol{d}, x_j, \boldsymbol{r}_\beta)$ into the numerator and denominator and separately Newton-reconstruct them individually in x_j from the set of sample points $x_{j,\alpha}$ with $\alpha \in \{1, \dots, N_j\}$,

$$F_{d,x_j}(\boldsymbol{d}, x_j, \boldsymbol{r}_\beta) = \frac{\text{Newton}_{x_j}[\text{Numerator}[V(\boldsymbol{d}, x_j, \boldsymbol{r}_\beta)], N_j]}{\text{Newton}_{x_j}[\text{Denominator}[V(\boldsymbol{d}, x_j, \boldsymbol{r}_\beta)], N_j]} \tag{19}$$

7. Proceed to step 3 for the next variable x_{j+1}. If $j = L$, the reconstruction stops.

Having introduced the algorithm, let us proceed with its optimization.

4. Optimal ordering

The order of the balanced reconstruction for multivariate functions is crucial for the overall execution speed of the algorithm as it affects the number of required black box probes P for a robust recovery. Thus, the minimization of the value of P serves as the main criterium for optimization. As a rough, naive estimate, it suffices to use the following rule of thumb: start with the variables requiring the lowest number of Thiele samples $T_j = \min\{T_k, k = 1, \dots, L\}$ and then proceed in the order of their growth (though, their order is not very relevant)

$$P_{\text{naive}} \simeq T_j \prod_{k \neq j}^{L} [T_k/2] . \tag{20}$$

However, a more accurate value of P was found experimentally by minimizing the product

$$P_{\text{balance}} = \min \left\{ N_L \Big[N_{L-1} \Big[\dots \big[N_2(N_1 + D_1) + D_2 \big] \dots \Big] + D_{L-1} \Big] + D_L \right\} , \tag{21}$$

where

$$D_j = \max(T_j - N_j, 0) , \tag{22}$$

expressed in terms of the minimal number of the Thiele (8) and balanced Newton (13) sampling points, respectively. While the value of T is self-explanatory, N is determined from

$$N \simeq \max\left\{\deg_x[\text{Numerator}[f(x)]], \deg_x[\text{Denominator}[f(x)]]\right\} + 2\,, \tag{23}$$

as it is clear from Eq. (19) with the addition of 2 needed for restoration of an overall constant and control probe for correctness of recovery. A rigorous proof of the above estimate (21) would be very welcome.

The number of different combinations one has to compare in order to determine (21) is set by the number L of components in $\boldsymbol{x}$ and equals to the number of inequivalent permutations $L!$. Even for $L = 10$, it is the minuscule $3,628,800$ compared to the staggering, e.g., 11 trillion operations per second that, for instance, an M1 Max silicone can perform.

5. Benchmark and comparison

As a benchmark, let us confront our new approach with the method of multivariate homogeneous interpolation [21,22], and its reincarnation relevant to the dense rational reconstruction in Ref. [25], and the Zippel algorithm [24] implemented in [26].

The main idea behind the method of Refs. [21,22] consists in the homogeneous rescaling of all components of the vector $\boldsymbol{x}$ and introduction of a new function $h(z, \boldsymbol{x})$

$$\boldsymbol{x} \to z\boldsymbol{x}\,, \qquad h(z, \boldsymbol{x}) = F(z\boldsymbol{x})\,, \tag{24}$$

such that one can clearly separate its numerator P from the denominator Q. The algorithm then consists in just three steps.

1. Thiele reconstructs the variable z with T_z black box probes (8) for arbitrary fixed values of all other variables $\boldsymbol{x}_0$.
2. Separate its numerator $P(z\boldsymbol{x}_0)$ and the denominator $Q(z\boldsymbol{x}_0)$ and Newton reconstruct them separately in x_j variable on N_j sample points given in Eq. (23).
3. Proceed to step 2 for the next variable x_{j+1}. If $j = L - 1$, the algorithm stops.

A few of comments are in order. First, the method requires generalization of the original function by introducing a new variable. Then, however, there is no need to reconstruct the last variable x_L since it can be recovered using homogeneity. Second, there is an unpleasant subtlety in its application to denominators not possessing a constant term, which then vanishes for the point $\boldsymbol{x} = 0$, and the rational function becomes singular. If this is the case, one has to perform ad hoc shift of all variables [21,22] and only then apply the above algorithm. This could potentially result in a more elaborate reconstruction process though. Third, the advantage of this method is the complete democracy among different ordering of variable reconstructions, there is not a preferred one as compared to our balanced algorithm that we advocated for above. This immediately provides an estimate on the number of black box probes required for the robust reconstruction, cf. (20),

$$P_{\text{homogeneous}} \simeq T_z \prod_{k}^{L-1} N_k\,. \tag{25}$$

The Zippel algorithm [24] is advantageous when dealing with sparse polynomials, see, in particular, discussion in Section 2.1 of the first paper in Ref. [26]. Like in univariate techniques alluded to in Section 2 earlier, this method interpolates one variable at a time. At each step (aka

Table 1
Comparison between the numbers of sample points required for the balanced, homogeneous and Zippel reconstructions.

Rational function	Balanced	Homogeneous	Zippel
$\frac{x_1}{x_1+x_2+x_3}$	64	36	25
$x_1x_2^2x_3^3+x_1x_2x_3+x_3+10$	67	156	42
$\frac{x_1^3+5x_1^2x_2^2+x_1x_3+x_3+1}{x_1+x_2+x_3^2+1}$	111	144	72

stage) coefficients of a newly found univariate polynomial are then interpolated as functions of the next variable. If a given coefficient does not arise at a given step, it is assumed to be zero at all subsequent stages as well. This obviously requires less black-box probes compared to dense interpolators. The estimate for the minimal number of sample points P_{Zippel} for a successful interpolation is not a priori known and requires practice runs.

Comparing $P_{\text{homogeneous}}$ with P_{balance}, one immediately observes that while the number of sample points for the homogenous reconstruction depends on the cumulative power of the function in question, i.e., its proportionality to T_z, the one for the balancing method is controlled by the individual powers of each variable. In other words, if the function possesses a very high total power while the individual exponents of variables building it up are small, the balancing method will be far more effective compared to its homogenous counterpart. The Zippel algorithm is undoubtedly superior to both for sparse polynomials, as can be easily seen from Table 1[4].

Let us provide now asymptotic estimates for the number of black box probes for both methods as the number of variables tends to infinity. Introducing the maximal exponent p_j of each x_j variable in the rational function $F(\boldsymbol{x})$ as

$$p_j = \max\left\{\deg_{x_j}[\text{Numerator}[F(\boldsymbol{x})]], \deg_{x_j}[\text{Denominator}[F(\boldsymbol{x})]]\right\}, \tag{26}$$

we will use $p_m = \max_j\{p_j\}$ as their upper limit estimate. Then, according to Eq. (21)

$$P_{\text{balance}} \sim \mathcal{O}(p_1 \dots p_L) \leq O(p_m^L). \tag{27}$$

On the other hand, the homogeneous reconstruction requires $T_z \sim \mathcal{O}(p_1 + \cdots + p_L) \leq O(Lp_m)$ samples on the first step, with the other $L-1$ variables requiring $\mathcal{O}(p_m)$ probes. Cumulatively, this gives

$$P_{\text{homogeneous}} \leq O(Lp_m^L). \tag{28}$$

Thus, the balancing method is advantageous to the homogeneous one since, in spite of a more complex organization of the algorithm, it requires less sample points for a robust multivariate reconstruction, especially with the growth of L. Its obvious disadvantage is the requirement for establishing a proper reconstruction order, which can however be easily achieved by means of preparatory estimates for each of the variable involved. And these are not time consuming.

6. Code `Reconstruction.m` and integration with FIRE

The algorithm introduced in Sect. 3 was implemented as a Mathematica code `Reconstruction.m` and integrated within the FIRE6 environment [5] for IBP reduction of Feynman

[4] We would like to thank the anonymous referee for providing the estimates for the Zippel algorithm.

integrals. The code is attached with this submission and can be simply copied into the already existing `fire/FIRE6/mm/` folder of FIRE6 installation. Alternatively, it is freely distributed via the repository

```
https://bitbucket.org/feynmanIntegrals/fire/src/master/FIRE6/mm/Reconstruction.m
```

The main component of FIRE6 used as a input for the code is its modular arithmetic output obtained with its `FIRE6p` binary to generate IBP tables `filename_x1_.._xL_p.tables`. The file names imply that one chooses fixed numerical values for all variables, i.e., space-time dimension and Lorentz invariants, in the field of integer numbers modulo `p` with the value of `p` being the index of a set of hard-coded primes close to 2^{64}. It is chosen with the `#prime` option in FIRE. The main reason to work with modular rather than integer arithmetic directly, is that the former is easier compared to the latter since there are only finitely many elements to deal with, as we explained at length in the Introduction, so that to find a solution to a given problem one could try every possibility.

The first order of business is to perform the inverse transformation from the field of primes to rationals since sample information over distinct fields can be combined together with the help of the Chinese remainder algorithm [14]. It is accomplished with the command

```
RationalReconstructTables["filename_x1_..._xL_p.tables",prime_max]
```

where the syntax is self-explanatory and `prime_max` stands for the maximal value of `p`’s used (starting from 1). The output is the tables `filename_0.tables`.

Next, the first variable `x1` is reconstructed with the Thiele method using the command

```
ThieleReconstructTables["filename_x1_..._xL_0.tables",
                        x1->Range[x1_min, x1_max]]
```

from its range `Range[x1_min, x1_max]`.

Analytic dependence on the remaining variables, say `x2`, is found by means of the balanced Newton command

```
BalancedNewtonReconstructTables["filename_x1_x2_..._xL_0.tables",
                                x2->Range[x2_min, x2_max],x1->x1_0]]
```

for a fixed value `x1_0` of the done variable, which was used to prepare the balancing tables, and `x2` reconstructed from its values in the range `Range[x2_min, x2_max]`. The process is then repeated for the other $(L-2)$ `x`’s.

To provide more input on the syntax of these commands, we will turn to an example in the next section along with a thorough discussion of the optimization of the reconstruction order.

7. Examples

Since there is no essential time-wise difference for the modular component of FIRE6 to handle multiloop Feynman integrals, we choose to demonstrate details of the reconstruction procedure with a planar double box for two kinematical settings, which result in two and three kinematical invariants, respectively. A user-friendly Mathematica notebook accompanies this manuscript as an ancillary file along with all required scripts. All computations were done on a 10 core MacBook Pro with Apple M1 Max silicone and 64 GB RAM.

7.1. Three-variable reconstruction

We start with a massless double box, which is parametrized by two Mandelstam variables $s = -2p_1 \cdot p_2$ and $t = -2p_1 \cdot p_3$ and the space-time dimension d, such that $\boldsymbol{x} = (d, s, t)$.

7.1.1. Preparation and estimates

We begin with a preparation of the start file for the IBP reduction by running it in Mathematica:

```
Get["FIRE6.m"];
Internal={k1,k2};
External={p1,p2,p3};
Propagators={-k1^2,-(k1+p1+p2)^2,-k2^2,-(k2+p1+p2)^2,-(k1+p1)^2,-(k1-k2)^2,
             -(k2-p3)^2,-(k2+p1)^2,-(k1-p3)^2};
Replacements={p1^2->0,p2^2->0,p3^2->0,p1p2->-s/2,p1p3->-t/2,p2p3->1/2(s+t)};
PrepareIBP[];
Prepare[AutoDetectRestrictions->True,LI->True,PositiveIndices->7];
SaveStart["doublebox"];
```

This creates `doublebox.start`.

Next, we need to get a good estimate for the minimal number of sample points required for each of the three variables involved. We create a configuration file with the content

```
#compressor         none
#threads            1
#fthreads           1
#variables          d,s,t
#start
#folder             directory/
#problem            1 doublebox.start
#integrals          doublebox.m
#output             doublebox.tables
```

where `doublebox.m` refers to a Mathematica script file with a set of initial Feynman integrals chosen for the IBP reduction and determination of an initial set of MIs. Even though, we would typically not recommend a user to employ initial integrals with nonvanishing powers of invariant scalar products (the last two entries of `Propagators`), it is not essential for our demonstration, so we create `doublebox.m` which contains a single integral `{1,{1,1,1,1,1,1,1,-1,-1}}`. Then, we run the bash script[5] (here for d)

```
#!/bin/bash
for d in {100..115}
do
  for p in {1..5}
```

[5] The syntax used holds for the private version of FIRE6, soon to be made available. For the current public version, the syntax is `FIRE6p -variables "$d"-"$s"-"$t"-"$p" -c doublebox -silent`.

```
  do
     FIRE6p -v "$d"_90_80_"$p" -c doublebox --quiet
  done
done
```

to find a set of tables in the format `doublebox_d0_90_80_p0.tables`. The rational reconstruction from the finite fields is then accomplished from these by executing the Mathematica command[6]

```
For[d0=100,d0<=115,++d0,RationalReconstructTables[
        "doublebox_"<>ToString[d0]<>"_90_80_0.tables",5,Silent->False]]
```

which generates the tables `doublebox_d0_90_80_0.tables` with reconstructed rational coefficients as well as messages in how many steps this was achieved. The one with the largest number, i.e., `Rational reconstruction stable after 2 steps` implies that we needed three primes to do it. Finally, an estimate on T_d is obtained with

```
ThieleReconstructTables["doublebox_d_90_80_0.tables",d->Range[100,115]]
```

This creates an output file `doublebox_d_90_80_0.tables` as well as a message `Thiele reconstruction stable after 11 steps`. The latter tells us that an unambiguous Thiele reconstruction required $T_d = 12$ tables. Similar consideration is then performed for the other two variables and we conclude the following: three primes are needed for the rational reconstruction of both s and t and $T_{s,t} = 6$ is the minimal number of tables for their rational Thiele reconstruction.

7.1.2. Rational reconstruction and Thiele

To start the actual reconstruction process, we need to create IBP tables making use of the above estimates for the minimal number of data points in each variable. These numbers depend on the order in which the recovery sequence is performed. Without any attempt to optimize it at this stage (we will dwell on it later in Sect. 7.1.4), let us consider $d - s - t$ ordering. That is, we start with the variable d and use $T_d = 12$ as a minimal number of samples in this variable, since it will be recovered with the Thiele method, while the remaining two will be reconstructed by means of the balanced Newton and these require only about half of the data points $N_{s,t} \simeq T_{s,2}/2$. Also to warrant a robust rational restoration from primes and thus to be on a safe side, we add[7] an extra prime as well, i.e., we change the maximal value of `p` from 4 to `prime_max` = 4. As it is obvious from the naive estimate of P_{naive} in Eq. (20), $d - s - t$ ordering is one of the inefficient routes.

After running the script

```
#!/bin/bash
for t in {80..83}
do
```

[6] There is no need to load `Reconstruction.m` (separately from `FIRE6.m`) before this evaluation as it is intrinsically integrated in FIRE6.

[7] If one wants a faster performance, one could forego this increase.

```
  for s in {90..93}
  do
    for d in {100..111}
    do
      for p in {1..4}
      do
        FIRE6p -v "$d"_"$s"_"$t"_"$p" -c doublebox --quiet
      done
    done
  done
done
```

we generate a large list of tables `doublebox_d0_s0_t0_p0.tables` with fixed integer values `d0, s0, t0, p0` of all variables in their respective ranges. The rational reconstruction from primes is done with the Mathematica command

```
For[t0=80,t0<=83,++t0,For[s0=90,s0<=93,++s0,For[d0=100,d0<=111,++d0,
RationalReconstructTables["doublebox_"
<>ToString[d0]<>"_"<>ToString[s0]<>"_"<>ToString[t0] <>"_0.tables",4]]]]
```

and results in `doublebox_d0_s0_t0_0.tables`, with subsequent restoration of the variable d via the Thiele method

```
For[t0=80,t0<=83,++t0,For[s0=90,s0<=93,++s0,
ThieleReconstructTables["doublebox_d_"
<>ToString[s0]<>"_"<>ToString[t0]<>"_0.tables",d->Range[100,111]]]]
```

7.1.3. Balancing and balanced Newton

Next, we turn to the balanced Newton reconstruction of the variable s. To this end, we have to first create its balancing tables. This is done for a single value of the already recovered variable d (below `d0=100`), however, for the entire range $\alpha \in \{1, \dots, N_t\}$ of values t_α of the variable t (the very same ones as used in the construction of the initial tables in Sect. 7.1.2) but a wider range $\beta \in \{1, \dots, T_s\}$ of values s_β for the variable s in order to be able to restore it by mean of the Thiele method (see Sect. 7.1.1). Thus, we run the script

```
#!/bin/bash
for d in 100
do
  for t in {80..83}
  do
    for s in {90..95}
    do
      for p in {1..4}
      do
        FIRE6p -v "$d"_"$s"_"$t"_"$p" -c doublebox --quiet
      done
    done
```

```
  done
done
```

with subsequent rational

```
For[t0=80,t0<=83,++t0,
For[s0=90,s0<=95,++s0,RationalReconstructTables["doublebox_100_"
<>ToString[s0]<>"_"<>ToString[t0]<>"_0.tables",4]]]
```

and Thiele reconstructions

```
For[t0=80,t0<=83,++t0,ThieleReconstructTables["doublebox_100_s_"
<>ToString[t0]<>"_0.tables",s->Range[90,95]]]
```

The latter are now the *balancing tables* for the variable s that we sought for. Now calling the Mathematica command

```
For[t0=80,t0<=83,++t0,BalancedNewtonReconstructTables["doublebox_d_s_"
<>ToString[t0]<>"_0.tables",s->Range[90,93],d->100,Silent->False]]
```

we completely reconstruct the s-dependence (in addition to the previously restored d-dependence).

Following the very same steps all over again but now for t, we recover it as well. The output is a file `doublebox_d_s_t_0.tables` with full analytical dependence on all variables involved. In order to avoid being repetitive, we relegate our reader's curiocity to the accompanying Mathematica notebook for details.

7.1.4. Optimization

Finally, let us address the question of the most optimal choice for the variables' sequence during the restoration process: these are not created equal. In spite of the fact that the time for computation of individual probes is about 2.5 seconds,[8] when many samples are needed, the total time it takes to compute the initial set of black box probes can get large since the growth is linear. We conducted a numerical experiment to verify that the time reduction factors are linearly correlated with the total number of probes required for a robust reconstruction. This was indeed confirmed and is reported in Table 2.

7.2. Four-variable reconstruction

Let us move on to considering a case of the next level of difficulty, involving four variables. To stay with the same topology as above, we take one of the four external legs off-shell, say $p_4^2 \neq 0$. This yields three independent kinematical invariants, which we conveniently define in a symmetric fashion as $u = -2p_1 \cdot p_2$, $v = -2p_2 \cdot p_3$ and $w = -2p_1 \cdot p_3$. Together with the space-time dimension d, now $\boldsymbol{x}$ is (d, u, v, w). Since there are no conceptual or algorithmic differences in the four- (or more) variable case(s) and in order to spare the reader from parroting previous sections, we relegate all details to the accompanying Mathematica notebook `reconstruction.nb` and accompanying script files. Here, we merely content ourself with a general

[8] With the public version of FIRE6, this time is about 20 seconds.

Table 2
Comparison of 3! choices for the order of functional reconstruction against the number of table required according to Eq. (21) and numerical experiments yielding corresponding time reduction factors.

Order	Min. number of tables	Time reduction factor
$d-t-s$	907	1
$d-s-t$	878	0.96
$t-d-s$	854	0.91
$t-s-d$	778	0.83
$s-d-t$	582	0.67
$s-t-d$	573	0.66

comment. The actual reconstruction process takes just seconds, as in the previous example. What is more time-consuming now, on a machine with a small number of cores, is the preparation of modular tables, with time needed scaling linearly in the number of sample points in additional variables. Thus this portion of the complete reconstruction routine will heavily benefit from parallelization.

8. Conclusions

To conclude, in this exploratory paper, we introduced a new approach for robust reconstruction of rational functions of many variables from their modular arithmetic input. It is based on a balancing relation for recovery of a variable in question. The former is found from a small data set by means of the univariate Thiele method, which is then used in conjunction with the Newton reconstruction from a minimal original set of black box probes.

We developed a Mathematica language package, `Reconstruction.m`, which is intrinsically integrated into the FIRE6 environment for algebraic and modular arithmetic-based IBP reductions. We demonstrated its efficiency for a typical multiloop integral. We provided heuristic arguments for the most optimal choice of the multivariate reconstruction and confirmed them with numerical experiments.

While the presently suggested balanced method wins against the homogeneous reconstruction, it looses to the Zippel algorithm when applied to sparse polynomials since the latter requires far less black-box probes. The reason for this is that our method treats both dense and sparse functions on equal footings. Since the balanced method reconstructs one variable at a time, as the Zippel algorithm does as well, we can improve our method by explicitly enforcing the sparsity condition, i.e., the vanishing of certain expansion coefficients at all subsequent stages of reconstruction once they did not emerge at an early stage, and thus achieve a much more efficient framework. This integration of the Zippel algorithm into the balanced reconstruction will be implemented in a future version of the package.

A natural extension of the current work is to use it as a stepping stone for its C++ implementation along with addressing issues of optimization and parallelization for use on supercomputers.

CRediT authorship contribution statement

Belitsky, Smirnov, Yakovlev: methodology, software, writing and revisions.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Andrei Belitsky reports financial support was provided by National Science Foundation. Andrei Belitsky reports a relationship with National Science Foundation that includes: funding grants.

Data availability

No data was used for the research described in the article.

Acknowledgements

A.B. is deeply indebted to Vladimir Smirnov for his generous patience and meticulous explanations of inner workings of modern multiloop methods. We are grateful to Roman Lee for helpful correspondence and providing us with an updated version of LiteRed compatible with Mathematica 13.2. We also would like to thank the anonymous referee for a constructive feedback and making us vigilant regarding the Zippel algorithm. The work of A.B. was supported by the U.S. National Science Foundation under the grant No. PHY-2207138, while of A.S. and R.Y. by the Russian Science Foundation under the agreement No. 21-71-30003 (in part, for the development of a version of the algorithm applicable on supercomputers) and by the Ministry of Education and Science of the Russian Federation under the agreement No. 075-15-2022-284 as a part of the program at the Moscow Center for Fundamental and Applied Mathematics (in part, for the development of the balancing reconstruction method suitable for more that two variables).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.nuclphysb.2023.116253.

References

[1] K.G. Chetyrkin, F.V. Tkachov, Integration by parts: the algorithm to calculate beta functions in 4 loops, Nucl. Phys. B 192 (1981) 159–204.
[2] V.A. Smirnov, Analytic tools for Feynman integrals, Springer Tracts Mod. Phys. 250 (2012) 1–296.
[3] A.V. Smirnov, A.V. Petukhov, The number of master integrals is finite, Lett. Math. Phys. 97 (2011) 37–44, arXiv:1004.4199 [hep-th].
[4] C. Anastasiou, A. Lazopoulos, Automatic integral reduction for higher order perturbative calculations, J. High Energy Phys. 07 (2004) 046, arXiv:hep-ph/0404258 [hep-ph].
[5] A.V. Smirnov, F.S. Chuharev, FIRE6: Feynman integral REduction with modular arithmetic, Comput. Phys. Commun. 247 (2020) 106877, arXiv:1901.07808 [hep-ph].
[6] C. Studerus, Reduze-Feynman integral reduction in C++, Comput. Phys. Commun. 181 (2010) 1293–1300, arXiv:0912.2546 [physics.comp-ph];
A. von Manteuffel, C. Studerus, Reduze 2 - distributed Feynman integral reduction, arXiv:1201.4330 [hep-ph].
[7] P. Maierhöfer, J. Usovitsch, P. Uwer, Kira—a Feynman integral reduction program, Comput. Phys. Commun. 230 (2018) 99–112, arXiv:1705.05610 [hep-ph].
[8] J. Klappert, F. Lange, P. Maierhöfer, J. Usovitsch, Integral reduction with Kira 2.0 and finite field methods, Comput. Phys. Commun. 266 (2021) 108024, arXiv:2008.06494 [hep-ph].

[9] R.N. Lee, Presenting LiteRed: a tool for the loop InTEgrals REDuction, arXiv:1212.2685 [hep-ph];
R.N. Lee, LiteRed 1.4: a powerful tool for reduction of multiloop integrals, J. Phys. Conf. Ser. 523 (2014) 012059, arXiv:1310.1145 [hep-ph].

[10] S. Laporta, High precision calculation of multiloop Feynman integrals by difference equations, Int. J. Mod. Phys. A 15 (2000) 5087–5159, arXiv:hep-ph/0102033 [hep-ph].

[11] A. von Manteuffel, R.M. Schabinger, A novel approach to integration by parts reduction, Phys. Lett. B 744 (2015) 101–104, arXiv:1406.4513 [hep-ph].

[12] P.S. Wang, A p-adic algorithm for univariate partial fractions, in: Proc. ACM Symp. Symbolic Algebraic Comp. 1981, 1981, pp. 212–217;
P.S. Wang, M.J.T. Guy, J.H. Davenport, P-adic reconstruction of rational numbers, ACM SIGSAM Bull. 16 (1982) 2–3.

[13] M. Monagan, Maximal quotient rational reconstruction: an almost optimal algorithm for rational reconstruction, in: Proc. Int. Symp. Symbolic Algebraic Comp. ISSAC'04, 2004, pp. 243–249.

[14] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, Modern Computer Algebra, Cambridge University Press, 2013.

[15] M. Abramowitz, I.A. Stegun (Eds.), Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Dover Publications, 1964.

[16] M. Ben-Or, P. Tiwari, A deterministic algorithm for sparse multivariate polynomial interpolation, in: Proc. 20 Ann. ACM Symp. Theory of Computing 20th ACM STOC, 1988, pp. 301–309.

[17] R. Zippel, Interpolating polynomials from their values, J. Symb. Comput. 9 (1990) 375–403.

[18] D. Grigoriev, M. Karpinski, M.F. Singer, Computational complexity of sparse rational interpolation, SIAM J. Comput. 23 (1994) 1–11.

[19] E. Kaltofen, Greatest common divisors of polynomials given by straight-line programs, J. ACM 35 (1988) 231–264;
E. Kaltofen, B. Trager, Computing with polynomials given by black boxes for their evaluations: greatest common divisors, factorization, separation of numerators and denominators, J. Symb. Comput. 9 (1990) 301–320;
E. Kaltofen, Z. Yang, On exact and approximate interpolation of sparse rational functions, in: Proc. Int. Symp. Symbolic Algebraic Comp. ISSAC'07, 2007, pp. 203–210.

[20] J. de Kleine, M. Monagan, A. Wittkopf, Algorithms for the non-monic case of the sparse modular GCD algorithm, in: Proc. Int. Symp. Symbolic Algebraic Comp. ISSAC'05, 2005, pp. 124–131.

[21] A. Diaz, E. Kaltofen, FoxBox: a system for manipulating symbolic objects in black box representation, in: Proc. Int. Symp. Symbolic Algebraic Comp. ISSAC'98, 1998, pp. 30–37;
E. Kaltofen, W.-s. Lee, A.A. Lobo, Early termination in Ben-or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm, in: Proc. Int. Symp. Symbolic Algebraic Comp. ISSAC'00, 2000, pp. 192–201.

[22] A. Cuyt, W.-s. Lee, Symbolic and numerical algorithms sparse interpolation of multivariate rational functions, Theor. Comput. Sci. 412 (2011) 1445–1456.

[23] Q. Huang, X.S. Gao, Sparse polynomial interpolation with finitely many values for the coefficients, arXiv:1704.04359;
Q. Huang, X.S. Gao, Sparse rational function interpolation with finitely many values for the coefficients, arXiv:1706.00914.

[24] R.R. Zippel, Probabilistic algorithms for sparse polynomials, Lect. Notes Comput. Sci. 72 (1979) 216–226.

[25] T. Peraro, Scattering amplitudes over finite fields and multivariate functional reconstruction, J. High Energy Phys. 12 (2016) 030, arXiv:1608.01902 [hep-ph];
T. Peraro, FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs, J. High Energy Phys. 07 (2019) 031, arXiv:1905.08019 [hep-ph].

[26] J. Klappert, F. Lange, Reconstructing rational functions with FireFly, Comput. Phys. Commun. 247 (2020) 106951, arXiv:1904.00009 [cs.SC];
J. Klappert, S.Y. Klein, F. Lange, Interpolation of dense and sparse rational functions and other improvements in FireFly, Comput. Phys. Commun. 264 (2021) 107968, arXiv:2004.01463 [cs.MS].

[27] F.S. Chuharev, Development and implementation of multivariate rational reconstruction, in: Master Thesis [Advisor A. V. Smirnov], Moscow State University, 2022, pp. 1–25 (in Russian).

[28] M.S. Floater, K. Hormann, Barycentric rational interpolation with no poles and high rates of approximation, J. Numer. Math. 107 (2007) 315–331.